

Getting started with the STEVAL-3DP001V1 3D printer board

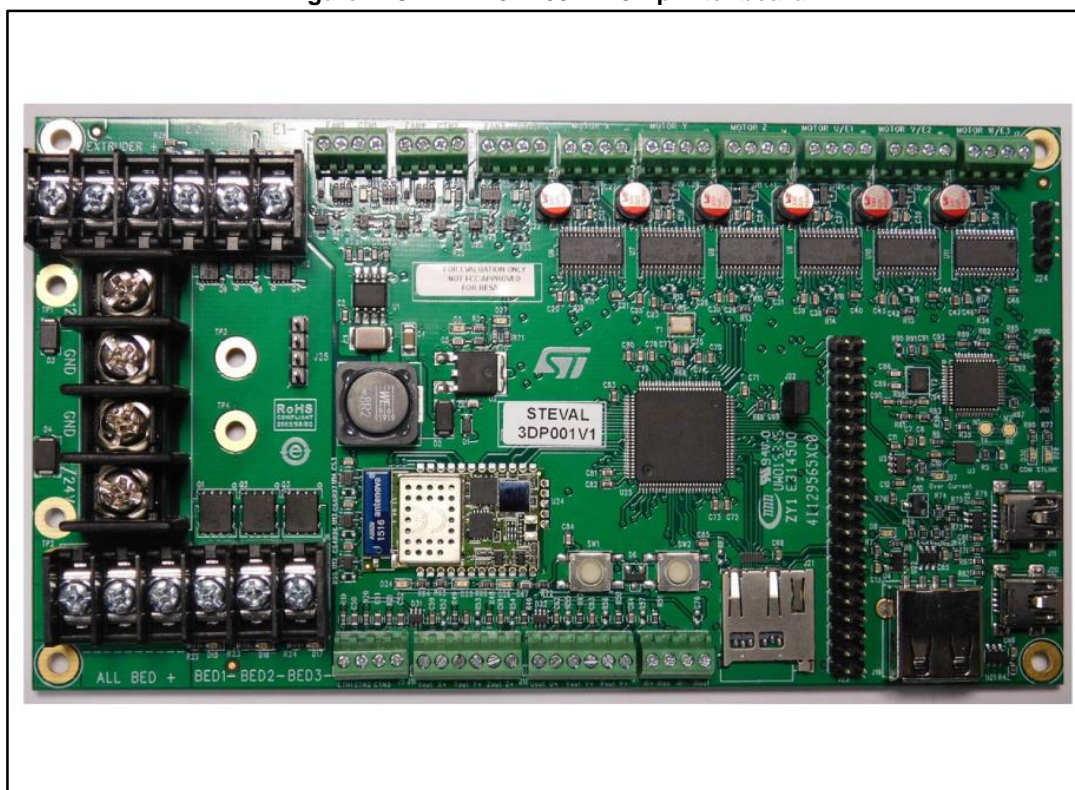
Introduction

The STEVAL-3DP001V1 is a complete and integrated solution for driving all 3D printers on the market, including Delta models requiring more complex computation. The solution is ideal for both beginners and experienced users. It is autonomous and can be used with a software interface or with custom firmware thanks to the embedded STM32 microcontroller based on the ARM 32-bit Cortex M4 core. The STEVAL-3DP001V1 is designed to drive 3D printers providing several axes (6 motors), several extruders (1 to 3), and multi-zone heating bed (1 to 3).

The STEVAL-3DP001V1 features integrated Wi-Fi connectivity, enabling the user to drive a 3D printer using a smartphone or tablet. The solution is also designed to work with 3D printer tools, such as Pronterface. USB connectivity is available through Virtual COM port, mini USB OTG and Dongle USB A. Moreover, the board includes a complete debug solution (STLINK-V2), a tool that is appreciated by developers.

The STEVAL-3DP001V1 allows connection to another board (e.g. Raspberry board or user board), with a connector that provides drive power (5 V - 3.3 V) and digital interface (SPI-I²C-ADC-GPIOS-SD-USB).

Figure 1: STEVAL-3DP001V1 3D printer board



Contents

1	Acronyms and abbreviations	6
2	Hardware.....	7
2.1	Power supply board (not included).....	7
2.2	Hardware description	8
2.4	Power supplies.....	9
2.4.1	5 V supply	9
2.4.2	3.3 V supply	9
2.5	Heated beds.....	9
2.6	Extruders.....	10
2.7	Axes	12
2.8	Connectivity.....	14
2.8.1	SPWF01SA Wi-Fi.....	14
2.8.2	MicroSD	14
2.8.3	USB OTG	14
2.8.4	USB Dongle.....	14
2.8.5	Switches	14
2.8.6	Expansion connector	14
2.9	Debug	15
3	Firmware	17
3.1	Firmware architecture	17
3.1.1	STM32Cube architecture.....	17
3.1.2	Marlin overview.....	18
3.1.3	Marlin4ST architecture	19
3.2	Firmware folder structure	19
3.2.1	BSP folders.....	20
3.2.2	Middleware folder	21
3.2.3	Project folder	21
3.3	Building and loading the firmware	21
3.3.1	Building the firmware with the OpenSTM32 IDE	21
3.3.2	Building the firmware with IAR IDE	25
3.3.3	Compilation flags	26
3.3.4	Loading firmware	26
3.4	Hardware resource mapping	27
3.5	Wi-Fi and web server	28

3.5.1	Loading the Wi-Fi firmware	28
3.5.2	Configuring the Wi-Fi module	28
3.5.3	Using the web pages	29
3.5.4	Customizing the web pages	34
3.6	Serial port	35
3.6.1	Printing via serial port	35
3.7	SD card	36
3.7.1	Configuration file	36
3.7.2	Printing from the SD	36
4	References	37
5	Revision history	38
Appendix A	OpenSTM32 installation	39

List of tables

Table 1: List of acronyms.....	6
Table 2: Power connector	9
Table 3: Heated bed connectors.....	10
Table 4: Extruders - head connectors.....	11
Table 5: Extruders - stepper motor connectors (U-V-W axes)	11
Table 6: End stop levels.....	12
Table 7: X-Y-Z steppers motors connections	12
Table 8: U-V-W steppers motor connections (extruders)	13
Table 9: End stop connections	13
Table 10: Expansion connector	15
Table 11: Document revision history	38

List of figures

Figure 1: STEVAL-3DP001V1 3D printer board	1
Figure 2: EVL400W-APD/ATX board.....	7
Figure 3: Jumpers and connectors	8
Figure 4: LED arrangement	8
Figure 5: Firmware architecture.....	17
Figure 6: Marlin4ST firmware architecture.....	19
Figure 7: OpenSTM32 – Eclipse IDE main window.....	22
Figure 8: OpenSTM32 – selecting import option	22
Figure 9: OpenSTM32 – selecting project to import	23
Figure 10: OpenSTM32 – generated binary files.....	24
Figure 11: OpenSTM32 – open Debug Configurations	24
Figure 12: OpenSTM32 – Debugger configuration.....	25
Figure 13: IAR IDE main window.....	26
Figure 14: axisctrl.shtml web page	30
Figure 15: command.shtml web page.....	31
Figure 16: heatctrl.shtml web page.....	31
Figure 17: extructrl.shtml web page.....	32
Figure 18: filemgt.shtml web page.....	33
Figure 19: wifictrl.html web page	34
Figure 20: Main window of Pronterface	36
Figure 21: Network configuration in Eclipse	39
Figure 22: Eclipse – available software selection	40
Figure 23: Eclipse – OpenSTM32 site information	40
Figure 24: OpenSTM32 AC6 tools installation	41

1 Acronyms and abbreviations

Table 1: List of acronyms

Acronym	Description
API	Application programming interface
BSP	Board support package
CMSIS	Cortex® microcontroller software interface standard
FW	Firmware
HAL	Hardware abstraction layer
IDE	Integrated development environment
LED	Light emitting diode

2 Hardware

The STEVAL-3DP001V1 integrates all the functions needed to drive most 3D printers on the market; it features:

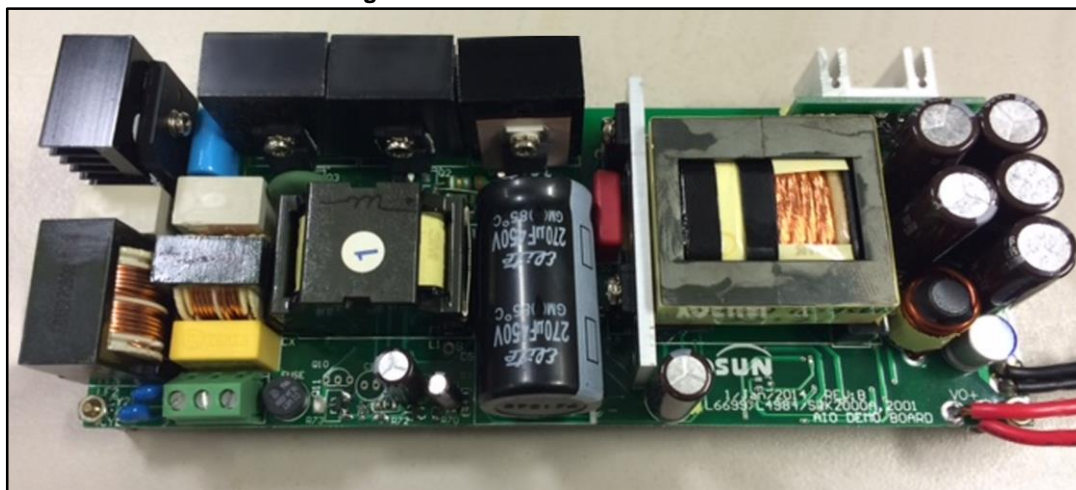
- Ready for the next generation of 3D printers
 - Multiple extruders support (up to 3) with temperature and fan control, LED display status, 12 V supply voltage and current capability up to 8 A
 - Multiple peripheral support and easy interfacing
 - Hot chamber and multi-zone heating bed with temperature control, LED display status, 12 V or 24 V supply voltage and current capability up to 20 A
- High efficiency and small footprint thanks to low $R_{DS(on)}$ MOSFETs
- Open source firmware available
- Main peripherals supported:
 - USB and microSD modules embedded
 - Wi-Fi module embedded with web server available
 - External LCD/keypad
- Based on STSPIN L6474 stepper motor driver with unique features in terms of current control and protection (able to drive up to 6 axes; phase current up to 3 A_{rms}; micro-stepping digital and end stop)
- Integrated debug solution (STLINKV2 embedded)
- RoHS compliant

2.1 Power supply board (not included)

The EVL400W-ADP/ATX board is recommended for supplying the 3D printer reference board. This SMPS provides 12 V 400 W regulated output from a universal 90 to 264 V_{AC} input voltage range.

All relevant information is available on www.st.com in the relevant [product page](#).

Figure 2: EVL400W-ADP/ATX board



2.2

Figure 3: Jumpers and connectors

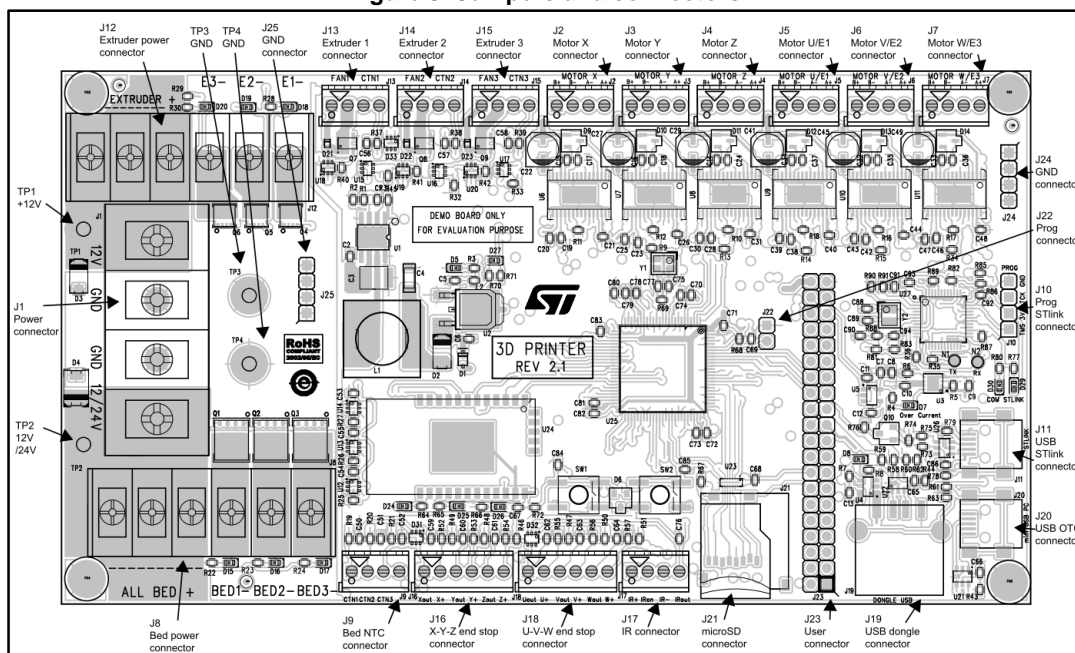
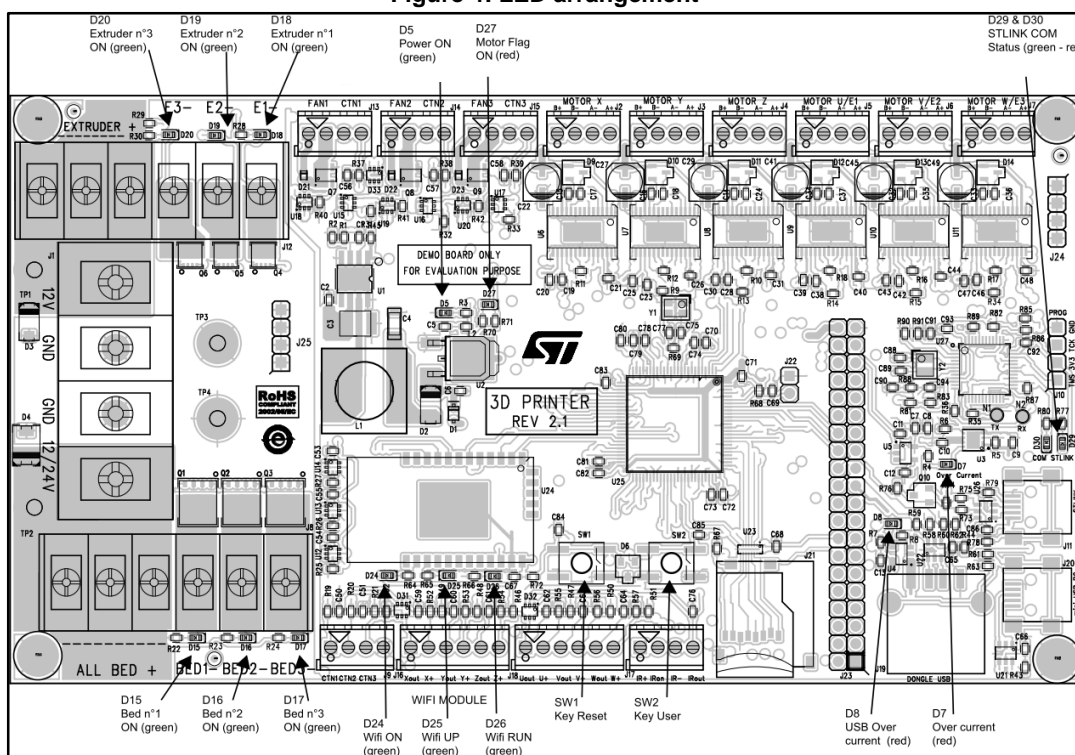


Figure 4: LED arrangement



2.4 Power supplies

The board is designed to be powered from two external supply voltages (see [Table 2: "Power connector"](#)):

The main supply powers all the board features except the heated beds (extruders, motors, 5 V regulator, 3.3 V regulator). The supply can be supplied between J1-1 and J1-2, or between TP1 and TP3 test points if more than 20 A is required for board operation.

The heated beds are supplied through a dedicated input supporting both 12 or 24 V, depending on the heated beds requirements. The supply is connected between J1-3 and J1-4, or between TP2 and TP4 test points for currents above 30 A.

Table 2: Power connector

Connector	Pin	Signal	Notes
J1	1	12 V	12 V main power supply (20 A max.)
	2	GND	GND for 12 V
	3	GND	GND for heated beds power
	4	12 / 24 V	Heated beds power supply (30 A max.)

2.4.1 5 V supply

The board generates a regulated 5 V supply for on-board circuitry, and user boards via the J23-37 pin.

5 V can be supplied through the ST-LINK USB instead of the external 12 V power supply, in which case the current is limited to 500 mA.

If LED D7 signals an overcurrent condition, disconnect the USB connector from J11.

2.4.2 3.3 V supply

The board generates a regulated 3.3 V supply for on-board digital (VDD_3V3) and analog (AVDD_3V3) circuitry, and user boards via the J23-40 pin.

LED D5 lights green when the 3.3 V supply is present.

2.5 Heated beds

The board can drive up to three heated beds with maximum 20 A each; all the beds are controlled independently.

The bed wires must be connected to the J8 connector; the thermistor (NTC) feedback for individual bed temperature control is connected to the J9 connector as per [Table 3: "Heated bed connectors"](#).

The recommended NTC value is 100 kΩ at 25 °C.

If one of the bed drivers is not used, the respective J8 and J9 connector lines can be left floating.

The status of each bed is signaled thus:

- D18 (red): heated bed 1 ON
- D19 (red): heated bed 2 ON
- D20 (red): heated bed 3 ON

Table 3: Heated bed connectors

Connector	Pin	Signal	Notes
J8	1	BED1+	12 / 24 V permanent (20 A max. each)
	2	BED2+	
	3	BED3+	
	4	BED1-	Open drain power switch
	5	BED2-	Open drain power switch
	6	BED3-	Open drain power switch
J9	1	NTC+ bed1	
	2	NTC+ bed2	
	3	NTC+ bed3	
	4	NTC- bed 1, 2, 3	GND

2.6 Extruders

The board can drive up to three fully independent extruders, with:

- a hot-end
- a 12 V fan
- a thermistor (NTC, 100 k Ω at 25 °C recommended)
- a stepper motor

The hot-end wires must be connected to the J12 connector (max. 8 A current each line) and the NTC temperature control feedback and fan driver for each extruder to J13, J14 and J15 connectors as per [Table 4: "Extruders - head connectors"](#).

The status of each hot-end is signaled thus:

- D15 (green): hot-end 1 ON
- D16 (green): hot-end 2 ON
- D17 (green): hot-end 3 ON

The printing material feeders (stepper motors) are connected to J5, J6 and J7 connectors as per [Table 5: "Extruders - stepper motor connectors \(U-V-W axes\)"](#).

If one of the extruders is not used, the corresponding connector line can be left floating or used for other purposes.

Table 4: Extruders - head connectors

Connector	Pin	Signal	Notes
J12	1	Extruder 1 hot-end+	12 V permanent (8 A max. each)
	2	Extruder 2 hot-end+	
	3	Extruder 3 hot-end+	
	4	Extruder 1-	Open drain power switch
	5	Extruder 2-	Open drain power switch
	6	Extruder 3-	Open drain power switch
J13	1	FAN+ Extruder 1	12 V permanent
	2	FAN- Extruder 1	Open drain power switch
	3	NTC- Extruder 1	GND
	4	NTC+ Extruder 1	
J14	1	FAN+ Extruder 2	12 V permanent
	2	FAN- Extruder 2	Open drain power switch
	3	NTC- Extruder 2	GND
	4	NTC+ Extruder 2	
J15	1	FAN+ Extruder 3	12 V permanent
	2	FAN- Extruder 3	Open drain power switch
	3	NTC- Extruder 3	GND
	4	NTC+ Extruder 3	

Table 5: Extruders - stepper motor connectors (U-V-W axes)

Connector	Pin	Signal	Notes
J5	1	Phase B+	Stepper motor of Extruder 1 - U axis (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	
J6	1	Phase B+	Stepper motor of Extruder 2 - V axis (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	
J7	1	Phase B+	Stepper motor of Extruder 3 - W axis (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	

2.7 Axes

The reference board can drive 3D printers with 3-axis positioning and up to 3 extruders or other devices, providing up to 6-axis positioning using the extruder stepper motors as extra axes.

For each axis, a digital end stop input can be set to active level high or low (see [Table 6: "End stop levels"](#)). The end stops are connected to J16 and J18 as per [Table 9: "End stop connections"](#).

The stepper motors driving each axis must be connected to the respective connector, as per [Table 7: "X-Y-Z steppers motors connections"](#) and [Table 8: "U-V-W steppers motor connections \(extruders\)"](#).

Table 6: End stop levels

End stop	Level of end stop	Resistor set up
X end stop level	High (default)	R49 = 0 / R52 = not populated
	Low	R49 = NP / R52 = 0
Y end stop level	High (default)	R48 = 0 / R53 = not populated
	Low	R48 = NP / R53 = 0
Z end stop level	High (default)	R46 = 0 / R54 = not populated
	Low	R46 = NP / R54 = 0
U end stop level	High (default)	R47 = 0 / R55 = not populated
	Low	R47 = NP / R55 = 0
V end stop level	High (default)	R50 = 0 / R56 = not populated
	Low	R50 = NP / R56 = 0
W end stop level	High (default)	R51 = 0 / R57 = not populated
	Low	R51 = NP / R57 = 0

Table 7: X-Y-Z steppers motors connections

Connector	Pin	Signal	Notes
J2	1	Phase B+	Stepper motor of axis X (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	
J3	1	Phase B+	Stepper motor of axis Y (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	
J4	1	Phase B+	Stepper motor of axis Z (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	

Table 8: U-V-W steppers motor connections (extruders)

Connector	Pin	Signal	Remarks
J5	1	Phase B+	Stepper motor of Extruder 1 - U axis (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	
J6	1	Phase B+	Stepper motor of Extruder 2 - V axis (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	
J7	1	Phase B+	Stepper motor of Extruder 3 - W axis (2 A / phase max.)
	2	Phase B-	
	3	Phase A-	
	4	Phase A+	

Table 9: End stop connections

Connector	Pin	Signal	Remarks
J16	1	X end stop output	
	2	X end stop level	
	3	Y end stop output	
	4	Y end stop level	
	5	Z end stop output	
	6	Z end stop level	
J18	1	U end stop output	
	2	U end stop level	
	3	V end stop output	
	4	V end stop level	
	5	W end stop output	
	6	W end stop level	

2.8 Connectivity

2.8.1 SPWF01SA Wi-Fi

Further information regarding the embedded SPWF01SA 802.11 b/g/n-compliant Wi-Fi module is available at www.st.com, including the [datasheet](#).

The SPWF01SA Wi-Fi status is signaled thus:

- LED D24 (green): power ON
- LED D25 (green): Link up
- LED D26 (green): Wi-Fi running

2.8.2 MicroSD

The board includes a microSD connector, compliant with SD Memory card specification V2.0, supporting 1- and 4-bit databus modes.

The signals for SD communication are also available on user connector J23, making it possible to add another SD card.

2.8.3 USB OTG

The board includes a USB OTG interface, available on the J20 Mini-B USB connector.



USB OTG Mini-B (J20) and USB dongle (J19) cannot be used simultaneously as they share the same USB interface.

2.8.4 USB Dongle

The board can accept a USB dongle via the J19 type-A USB connector.

The 5 V supply of the USB dongle connector is limited to 800 mA. If this value is exceeded, LED D8 (red) indicates an overcurrent condition. In this case, disconnect the USB dongle from the J19 connector.



USB OTG Mini-B (J20) and USB dongle (J19) cannot be used simultaneously as they share the same USB interface.

2.8.5 Switches

Reset switch SW1 resets all board CPUs and peripherals.

User switch SW2 is for user interaction; it can be configured in the firmware to trigger start/stop routines or other functions.

2.8.6 Expansion connector

The J23 40-pole connector allows extending the 3D printer reference board functionality with the connection of more hardware.

The following signals and peripherals are available on the connector (detailed pinout in [Table 10: "Expansion connector"](#)):

- Axes end stop signals
- Heated bed NTC feedback signals

- Heated bed PWM signals
- Fan PWM signal (see [Section 2.5: "Extruders"](#))
- SD card interface
- SPI interface
- UART interface
- I²C interface
- USB interface (shared between USB OTG and USB dongle)
- 4 GPIOs
- 5 V supply, 3.3 V supply and ground

Table 10: Expansion connector

Pin	Signal	Remarks	Pin	Signal	Remarks
1	W end stop	GPIO	2	V end stop	GPIO
3	U end stop	GPIO	4	Z end stop	GPIO
5	Y end stop	GPIO	6	X end stop	GPIO
7	NTC bed3	ADC	8	NTC bed2	ADC
9	NTC bed1	ADC	10	SD Dat1	SDIO
11	SD Dat0	SDIO	12	SD Clk	SDIO
13	SD CMD	SDIO	14	SD Dat3	SDIO
15	SD Dat2	SDIO	16	SD CD	SDIO
17	SPI MISO	SPI	18	SPI NSS	SPI
19	SPI SCK	SPI	20	SPI MOSI	SPI
21	V USB dongle	Power	22	USB OTG DM	USB
23	UART TX	UART	24	USB OTG DP	USB
25	UART RX	UART	26	I ² C SCL	I ² C
27	User 3	GPIO	28	I ² C SDA	I ² C
29	User 4	GPIO	30	Fan 3	GPIO
31	Bed Heat1	GPIO	32	Fan 2	GPIO
33	Bed Heat2	GPIO	34	Fan 1	GPIO
35	Bed Heat3	GPIO	36	User 1	GPIO
37	User 2	GPIO	38	VDD 5V	Power
39	GND	Power	40	VDD 3V3	Power

2.9 Debug

The board embeds an ST-LINK/V2-1 debugger/programmer. The features supported on ST-LINK are:

- USB software re-enumeration
- Virtual com port interface on USB
- Mass storage interface on USB
- USB power management request for more than 500 mA power on USB

The power supply for ST-LINK is provided either by the host PC through the USB cable connected to J11 or from the 12 V power supply (VDD_power). Both supplies must be present for ST-LINK debugging functions.

LEDs D29 (green) and D30 (red) provide ST-LINK communication status information:

- Red LED flashing slowly: at power-on before USB initialization
- Red LED flashing quickly: following first correct communication between the PC and ST-LINK/V2-1 (enumeration)
- Red LED ON: initialization between the PC and ST-LINK/V2-1 is complete
- Green LED ON: successful target communication initialization
- Red/Green LED flashing: during communication with target
- Green ON: communication finished and successful.

3 Firmware

3.1 Firmware architecture

The Marlin4ST firmware is the default firmware for the STEVAL-3DP001V1 3D printer board. It runs on the STM32F401 and is fully capable of handling 3D prints from G-Codes (Marlin format). The prints can be performed via the UART, SD or Wi-Fi interfaces.

It can be interfaced with 3D printer host software like Pronterface, Repetier Host and OctoPrint via UART.

The firmware comes with the complete source code for the OpenSTM32 (free) and IAR development environments.

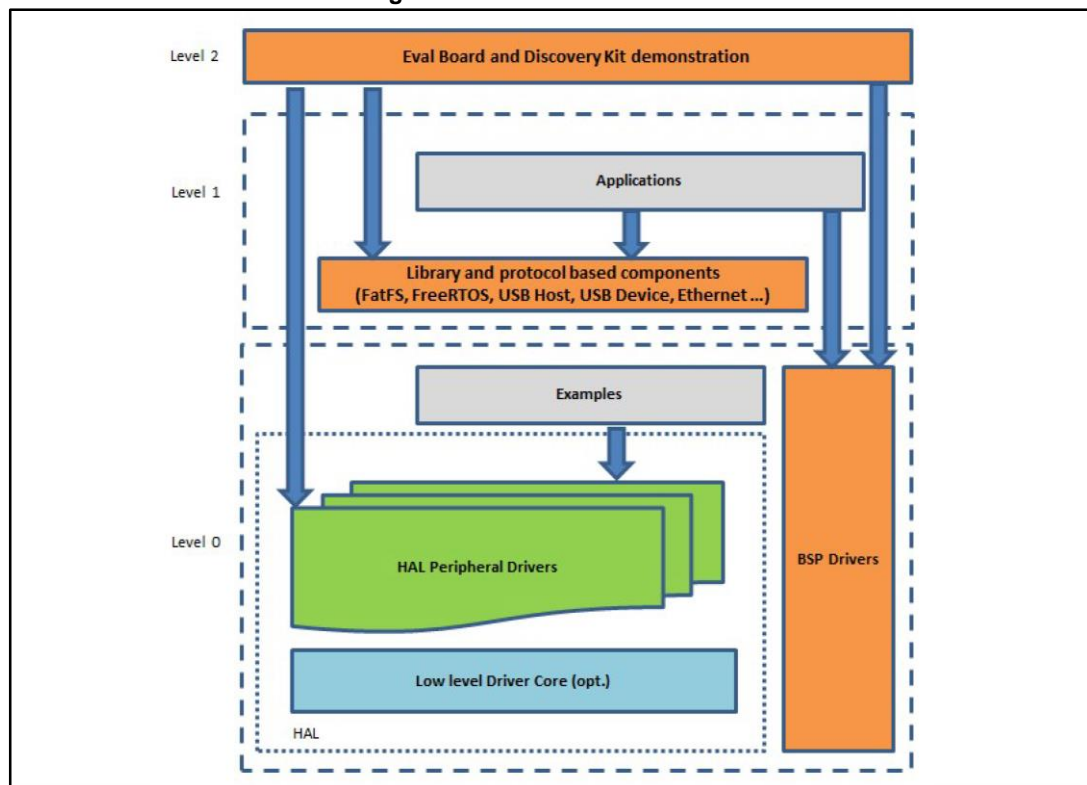
By default, it is configured to run on a Prusa I3 rework 5, but it can easily be set up to run on any FDM 3D printer.

The firmware is organized into a drivers section with STM32Cube microcontroller and peripheral drivers, a middleware section with STM32Cube FatFs and Marlin algorithms for motion control, G-Codes, etc., and an application section with user command entry points.

3.1.1 STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with one another, as described in the diagram below.

Figure 5: Firmware architecture



Level 0: This level is divided into three sub-layers:

- Board Support Package (BSP): this layer offers a set of APIs relative to the hardware components in the hardware boards (Audio codec, IO expander, Touchscreen, SRAM

driver, LCD drivers. etc...); it is based on modular architecture allowing it to be easily ported on any hardware by just implementing the low level routines. It is composed of two parts:

- Component: is the driver relative to the external device on the board and not related to the STM32, the component driver provides specific APIs to the external components of the BSP driver, and can be ported on any other board.
- BSP driver: links the component driver to a specific board and provides a set of easy to use APIs. The API naming convention is `BSP_FUNCT_Action()`: e.g., `BSP_LED_Init()`, `BSP_LED_On()`.
- It is based on modular architecture allowing it to be easily ported on any hardware by just implementing the low level routines.
- Hardware Abstraction Layer (HAL): this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). It provides generic, multi-instance and function-oriented APIs to help offload user application development time by providing ready to use processes. For example, for the communication peripherals (I²C, UART, etc.) it provides APIs for peripheral initialization and configuration, data transfer management based on polling, interrupt or DMA processes, and communication error management. The HAL Drivers APIs are split in two categories: generic APIs providing common, generic functions to all the STM32 series and extension APIs which provide special, customized functions for a specific family or a specific part number.
- Basic peripheral usage examples: this layer houses the examples built around the STM32 peripherals using the HAL and BSP resources only.

Level 1: This level is divided into two sub-layers:

- Middleware components: set of libraries covering USB Host and Device Libraries, STemWin, FreeRTOS, FatFS, LwIP, and PolarSSL. Horizontal interaction among the components in this layer is performed directly by calling the feature APIs, while vertical interaction with low-level drivers is managed by specific callbacks and static macros implemented in the library system call interface. For example, FatFs implements the disk I/O driver to access a microSD drive or USB Mass Storage Class.
- Examples based on the middleware components: each middleware component comes with one or more examples (or applications) showing how to use it. Integration examples that use several middleware components are provided as well.

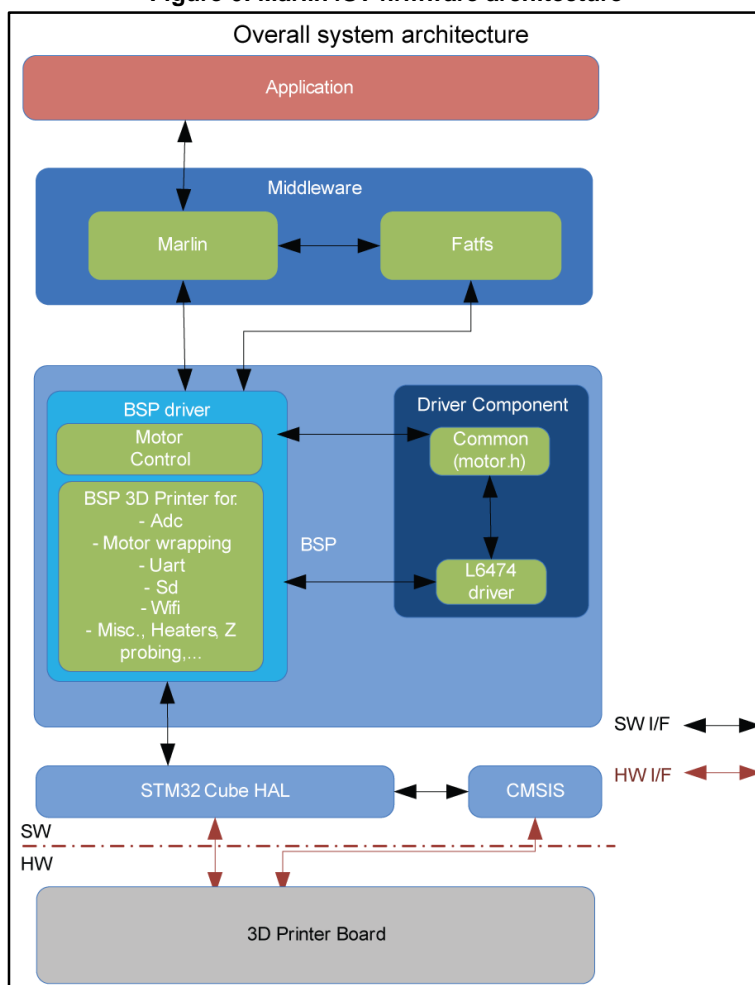
Level 2: This level is a single layer with a global, real-time and graphical demonstration based on the middleware service layer, the low level abstraction layer and basic peripheral usage applications for board-based functions.

3.1.2 Marlin overview

Information regarding the Marlin firmware by RepRap for the control of Arduino-based 3D printers is described here: <http://reprap.org/wiki/Marlin>. The code is available on GitHub: <https://github.com/MarlinFirmware/Marlin>.

3.1.3 Marlin4ST architecture

Figure 6: Marlin4ST firmware architecture



In accordance with STM32Cube, the Marlin4ST firmware has three main layers:

- Drivers: all the microcontroller and peripherals drivers directly from the STM32Cube environment.
- Middleware: integrates the FatFS from STM32Cube and the Marlin firmware with the 3D printer algorithm (motion control, G-Code parsing, temperature monitoring, etc.)
- Application: this layer is the main entry point of the firmware. The setup component initializes the system, while the infinite loop reacts to the commands that the user sent from the different interfaces (UART, SD, Wi-Fi).

3.2 Firmware folder structure

The Marlin4ST code package can be downloaded from Github:

<https://github.com/St3dPrinter/Marlin4ST>.

The code packaged in the following main folders:

- A Drivers folder with:
 - the required STM32Cube HAL files, located in the STM32F4xx_HAL_Driver subfolder. Only the STM32Cube framework HAL files required to run the 3D printer samples are included.

- a CMSIS folder with the CMSIS (Cortex® Microcontroller Software Interface Standard) vendor-independent hardware abstraction layer for the Cortex-M processor series. This folder is also derived directly from the STM32Cube framework.
- a board support package (BSP) folder with the code files required for the 3D printer board configuration, the L6474 driver and the motor control API.
- A Middleware folder with respective FatFs and Marlin folders.
- A Project folder with different IDE projects as well as the main.c/h files.

3.2.1 BSP folders

The 3D printer software uses the board support packages described below.

3.2.1.1 3D printer BSP

This BSP provides the interface to configure and use peripherals: ADC, SD, UART, Wi-Fi, SPI, etc. In the appropriate folder (stm32_cube/Drivers/BSP/STM32F4xx-3dPrinter), there are six source/header file pairs.

- stm32f4xx_3dprinter_adc.c/h: programming of the ADC and the associated DMA which is used to monitor heaters and bed temperatures with the associated thermistors.
- stm32f4xx_3dprinter_misc.c/h: configuration of the timers, the servo motor if used, the fans, etc.
- stm32f4xx_3dprinter_motor.c/h: configuration of the GPIOs, SPI and PWMs of the motor drivers
- stm32f4xx_3dprinter_sd.c/h: configuration of the SDIO and of the associated DMA
- stm32f4xx_3dprinter_uart.c/h: configuration of the UART ports available through the ST-LINK virtual com port.
- stm32f4xx_3dprinter_wifi.c/h: programming of the elements required to control the Wi-Fi module (UART, GPIOs, etc.)

3.2.1.2 Motor control BSP

This BSP provides a common interface to access the driver functions of various motor drivers like L6474, L648X, L647X, etc. This is done via the MotorControl/motorcontrol.c/h file pair, which defines the functions to configure and control the motor driver.

These functions are then mapped to the functions of the motor driver component used on the given expansion board via the motorDrv_t (defined in Components/Common/motor.h.) structure file which is written with a list of function pointers during instantiation in the corresponding motor driver component. For the 3D printer board, the instance is called l6474Drv (see stm32_cube/Drivers/BSP/Components/l6474/l6474.c file).

As the motor control BSP is common for all motor driver expansion boards, the unavailable functions for this board are replaced by null pointer during motorDrv_t instantiation in the driver component.

3.2.1.3 L6474 BSP component

The L6474 BSP component provides the driver functions for the L6474 motor driver in stm32_cube/Drivers/BSP/Components/l6474.

This folder has 3 files:

- l6474.c: core functions of the L6474 driver
- l6474.h: declaration of the L6474 driver functions and corresponding definitions
- l6474_target_config.h: predefines for the L6474 parameters and for the motor devices

3.2.2 Middleware folder

The middleware has the following components:

- FatFS deriving from the STM32 Cube environment without modification
- the Marlin adaptation of the Marlin firmware to the STM32 Cube environment.

The Marlin middleware is designed to minimize deviation from the original Marlin version to simplify updating to new Marlin versions. The files names are unchanged and only one “Marlin_export.h” file has been added to:

- wrap the low layer access between the Marlin and the Cube HAL (e.g., WRITE and READ macros).
- stub the unsupported or irrelevant Marlin functions in the STM32Cube environment

In the other Marlin middleware files, major changes are flagged under `STM_3DPRINT_2`.

3.2.3 Project folder

The 3D printer firmware can be built with the following IDEs:

- IAR: with project files in `stm32_cube\Projects\STM32F4xx-3dPrinter\Marlin\EWARM`
- OpenSTM32: with project files in `Projects\STM32F4xx-3dPrinter\Marlin\SW4STM32\Marlin`

The project folder also contains the application entry point files and certain configuration files:

- `Inc/ffconf.h`: FAT file system configuration file
- `Inc/main.h`: main header file
- `Inc/stm32f4xx_hal_conf.h`: AL configuration file for stm32f4 devices
- `Inc/stm32f4xx_it.h`: header for the interrupt handler for stm32f4 devices
- `Src/main.c`: main program (application entry point)
- `Src/stm32f4_hal_msp.c`: HAL initialization routines for stm32f4 devices
- `Src/stm32f4xx_it.c`: interrupt handler for stm32f4 devices
- `Src/system_stm32f4xx.c`: system initialization for stm32f4 devices
- `Src/clock_f4.c`: clock initialization for stm32f4 devices

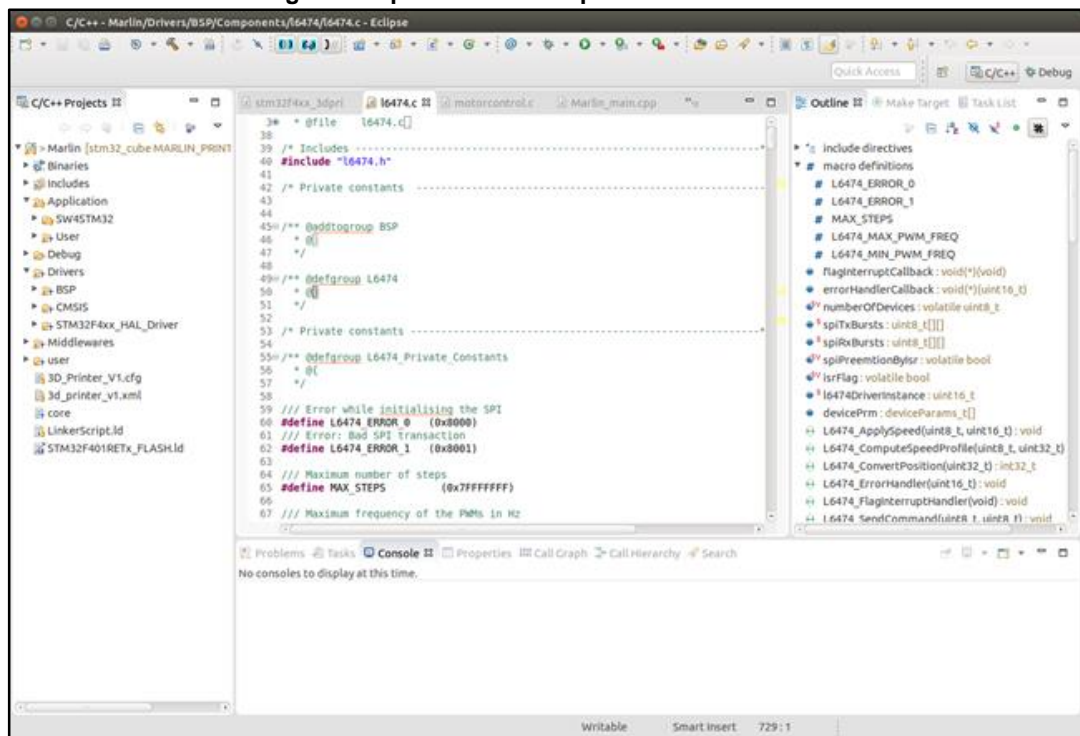
3.3 Building and loading the firmware

The following sections describe how to build the firmware in the OpenSTM32 and IAR development environments.

3.3.1 Building the firmware with the OpenSTM32 IDE

OpenSTM32 is based on the Eclipse IDE (refer to Appendix A for OpenSTM32 installation); the workspace is set to the root of the `stm32_cube` directory.

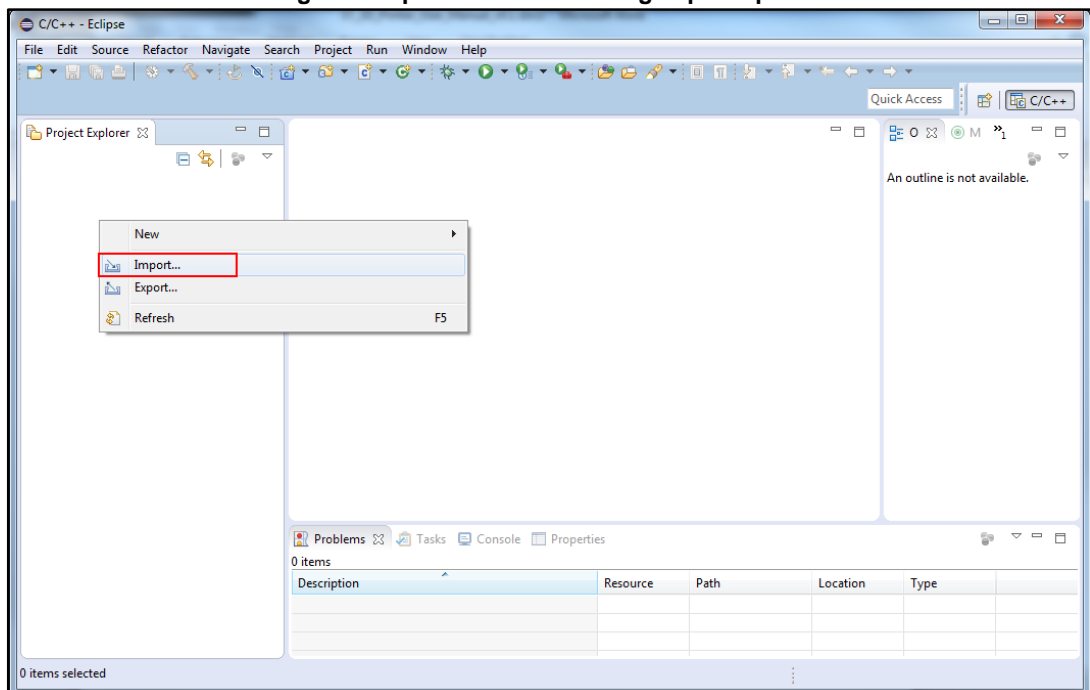
Figure 7: OpenSTM32 – Eclipse IDE main window



3.3.1.1 Import Marlin4ST project

STEP 1: to import the Marlin4ST project, right click in Project Explorer window and select Import.

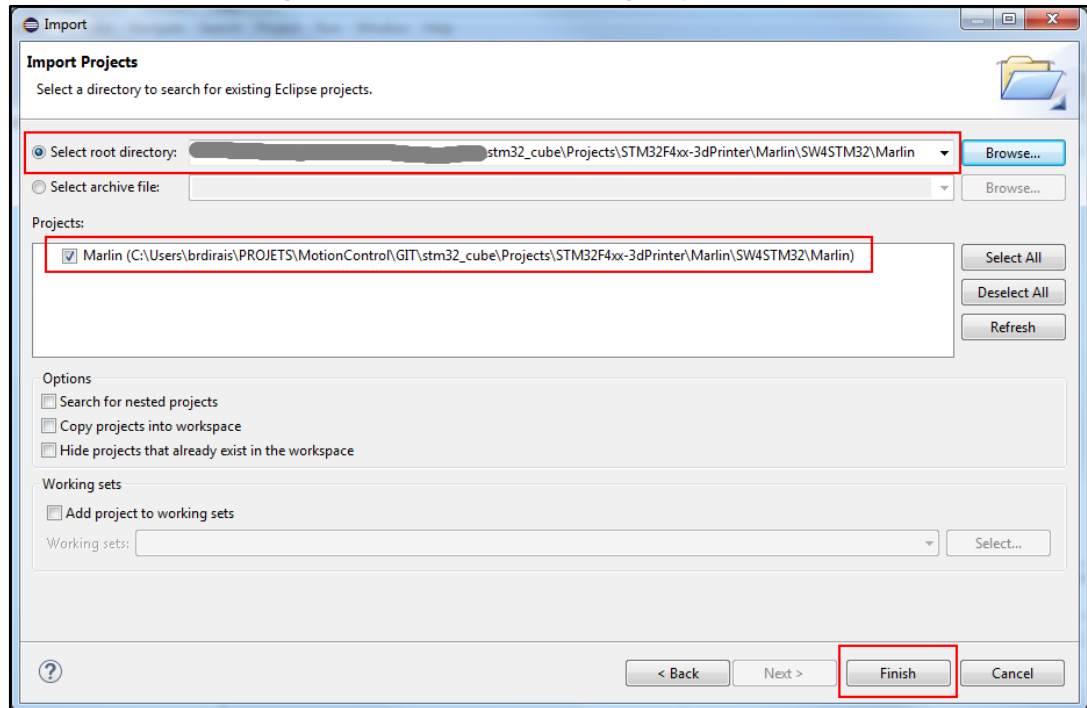
Figure 8: OpenSTM32 – selecting import option



STEP 2: a window appears; select General followed by Existing Projects into Workspace. This opens a new window where you can browse to select the Marlin4ST project in

ProjectsSTM32F4xx-3dPrinterMarlinSW4STM32Marlin. Select the appropriate project shown below and click Finish.

Figure 9: OpenSTM32 – selecting project to import



3.3.1.2 Build the binaries

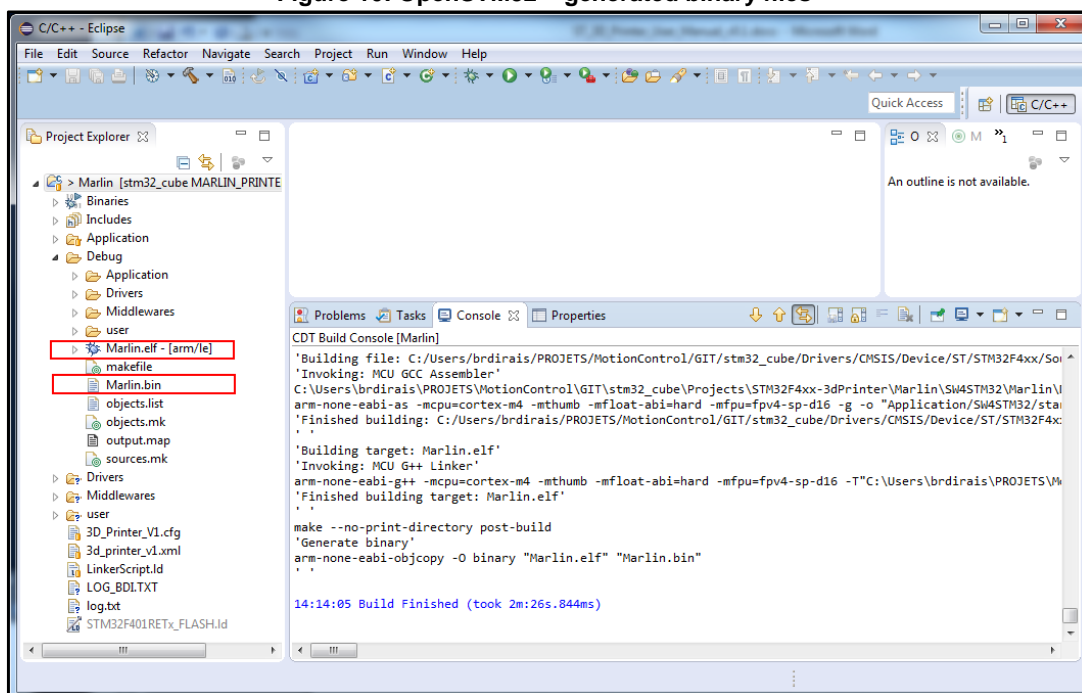
STEP 1: Right click on the project to display the menu.

STEP 2: select Clean Project followed by Build Project to compile your project.

Once compiled, the following files are available in the Debug or Release directory, according to the type of compilation you have chosen:

1. Marlin.elf: which can be loaded on the board using the OpenSTM32 tool,
2. Marlin.bin: which can be loaded through the mass storage interface (simply drag and drop to disk), as described in [Section 5.3.4: "Loading firmware"](#).

Figure 10: OpenSTM32 – generated binary files



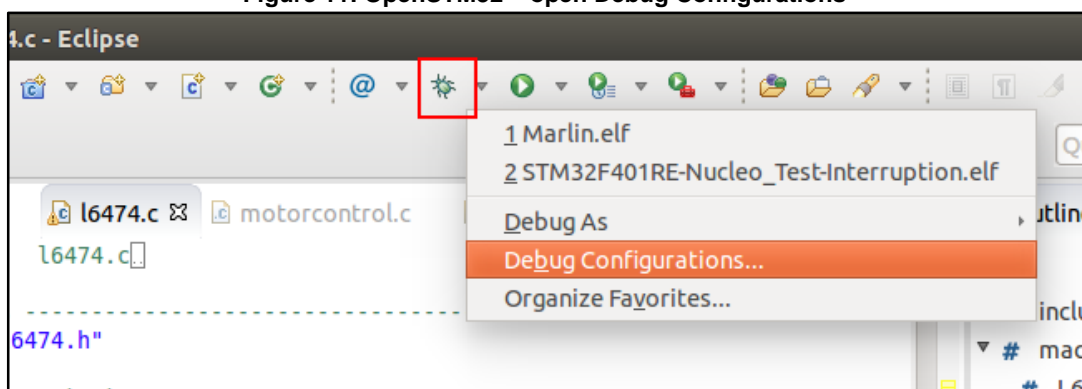
3.3.1.3 Debug project

Before starting debug, ensure that:

- your ST 3D printer board has been upgraded with the latest version of ST-LINK V2-1. You can search for the STSW-LINK007 firmware upgrade tool on www.st.com.
- a USB 2.0 port is used to connect your board (a problem has been identified with USB 3.0 and Windows 7.0 and above).

STEP 1: Open the Debug Configurations window, and select Marlin.elf in the Ac6 STM32 Debugging section.

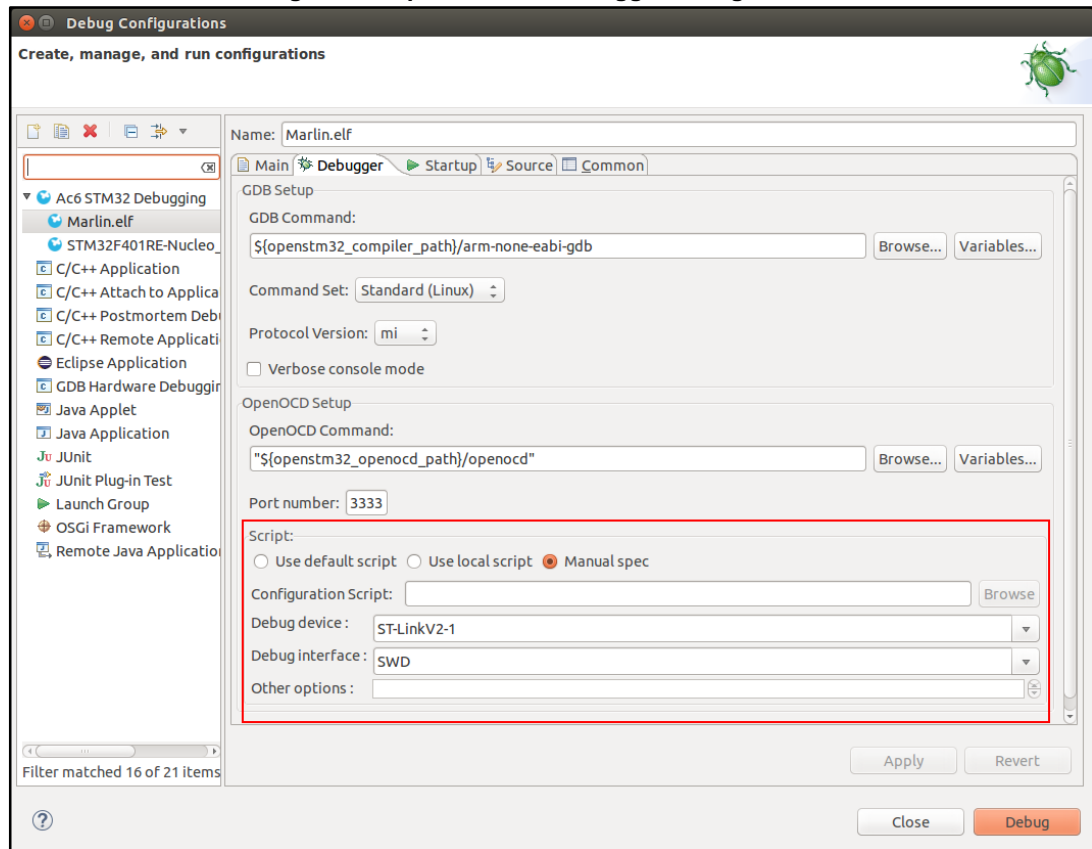
Figure 11: OpenSTM32 – open Debug Configurations



STEP 2: In the Script section of the Debugger tab, select Manual spec and ensure that:

- Debug device is set to ST-LinkV2-1
- Debug interface is set to SWD

Figure 12: OpenSTM32 – Debugger configuration



STEP 3: Start the debugging session by clicking the Debug button. Make sure that the board is connected to your computer via the ST-LINK V2-1 USB connector.

3.3.2 Building the firmware with IAR IDE

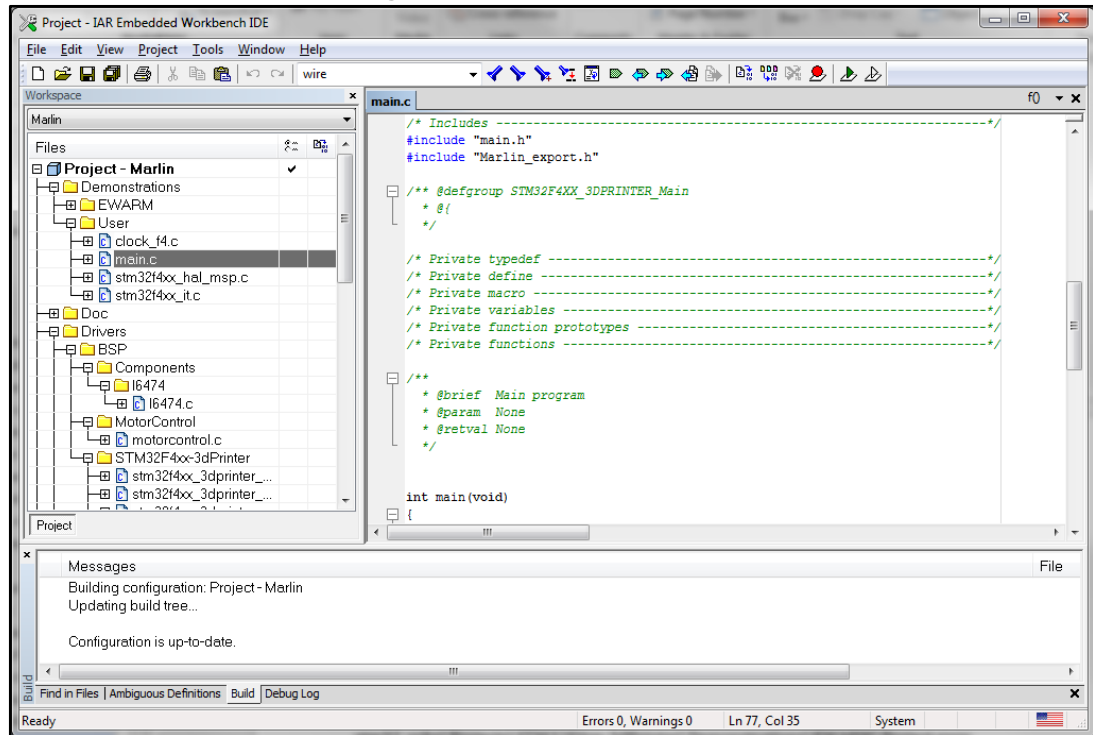
The Marlin4ST firmware also provides native support for the IAR Embedded Workbench IDE (<https://www.iar.com/iar-embedded-workbench/>).

The project is defined with an IDE version using:

- IAR Embedded Workbench for ARM v7.20.2.7431
- IAR Embedded Workbench common components v7.1.1.3263

STEP 1: to start the 3D printer project, simply open a workspace with:
 stm32_cube\Projects\STM32F4xx-3dPrinter\Demonstrations\EWARM\Project.eww

Figure 13: IAR IDE main window



STEP 2: build the project via the menu: Project > Rebuild All.

STEP 3: ensure the board is correctly supplied and a USB cable connects the printer board ST-LINK USB port and the PC and select “Download and Debug” from the menu to start debugging.

After the build command, you can also directly download STM32F4xx-3dPrinter\Demonstrations\EWARM\3DPrinter\Exe\Project.bin from the stm32_cube\Projects directory through the procedure described in [Section 5.3.4: "Loading firmware"](#).

3.3.3 Compilation flags

Whatever the build environment, you must set the following compilation flags to compile the Marlin4ST firmware:

- `USE_HAL_DRIVER`
- `STM32F401xE`
- `STM_3DPRINT`
- `MARLIN`

By default, these flags are already defined in OpenSTM32 and IAR project so no changes are required. You can declare the `NO_Wi-Fi` flag to disable the on-board Wi-Fi.

3.3.4 Loading firmware

The easiest way to load a binary into 3D printer board memory is to use the mass storage interface provided by the ST-LINK (simply drag and drop to disk), thus:

1. Connect the ST 3D printer reference board to a PC with a USB cable through the ST-LINK USB port. Jumper J22 (boot mode selection) must be set.
2. Power the 3D printer board via connectors J1 power connector (12/24 V and GND pins) to a DC power supply; a new drive should appear in your Explorer window.
3. Copy the binary file to the root of this new drive and refresh your file explorer: if the binary file has disappeared and no error log file has been generated, the binary file has been loaded successfully.
4. Reset the board to run the loaded binary file. To start correctly, the default firmware requires an SD card with a configuration file (see [Section 5.7.1: "Configuration file"](#)).

3.4 Hardware resource mapping

Motor step clock, dir, reset and flag pins; stops, servo, fans, heaters, beds and therm pins

The ST 3D printer board pins are defined in `stm32_cube\Middlewares\Third_Party\Marlin\pins.h`.

A positive value indicates that the pin is defined, is on the board and can be set or reset by the Marlin middleware using WRITE and READ macros defined in `stm32_cube\Middlewares\Third_Party\Marlin\Marlin_export.h`.

A minus one (-1) value indicates that this pin is either not on the board or not directly used by the Marlin middleware (it may be used by the BSP drivers).

In `stm32_cube\Drivers\BSP\STM32F4xx-3dPrinter\stm32f4xx_3dprinter_misc.c`, the arrays `gArrayGpioPort` and `gArrayGpioPin` are declared and initialized with "0" or with definitions. Each pin defined with a positive value in `pins.h` is an index in these two arrays and the definitions corresponding to this index are the port and corresponding pin number, respectively. These definitions are defined in files, depending of their meanings:

- `stm32f4xx_3dprinter_motor.h` for axis and extrusion motor step clocks, direction, reset and flag pins
- `stm32f4xx_3dprinter_misc.h` for stops, heaters and servo pins
- `stm32f4xx_3dprinter_adc.h` for thermistor pins

These two arrays have definitions for some indices which do not correspond with positive value pins defined in `pins.h`. This is the case for pins that are only used directly by the BSP drivers.

For example, at index 2 of the `gArrayGpioPort` array in `stm32f4xx_3dprinter_misc.c`, the `BSP_MOTOR_CONTROL_BOARD_RESET_X_PIN` definition finds no corresponding value of 2 in `pins.h` as the `BSP_MOTOR_CONTROL_BOARD_RESET_X_PIN` is only used in `BSP_MotorControlBoard_GpioInit`, `BSP_MotorControlBoard_ReleaseReset` and `BSP_MotorControlBoard_ReleaseReset` functions.

SPI pins

The SPI pins used for communication between the microcontroller and the motor driver circuits are defined in `stm32f4xx_3dprinter_motor.h`.

A second SPI is available to the user, with the corresponding pins defined in `stm32f4xx_3dprinter_misc.h`.

SD card pins

The SD detection pin is defined in `stm32f4xx_3dprinter_sd.h`. Only one SDIO set of pins is available on the STM32F401VE.

Wi-Fi pins

The Wi-Fi pin definitions are in `stm32f4xx_3dprinter_wifi.h`; they are only used directly by the BSP Wi-Fi driver.

User available resources

Four GPIOs, one I²C and one SPI are available to the user, with definitions set in `stm32f4xx_3dprinter_misc.h`.

3.5 Wi-Fi and web server

3.5.1 Loading the Wi-Fi firmware

The SPWF01SA ST Wi-Fi module is already loaded with firmware version SPWF01S-150410-c2e37a3. You can load new firmware over the air (FOTA) with a PC with Wi-Fi running Windows 7, a web server and a serial port terminal.

FOTA functionality has been tested to work with the following software versions:

- Apache server windows install `httpd-2.2.25-win32-x86-openssl-0.9.8y.msi` (<https://archive.apache.org/dist/httpd/binaries/win32/>)
- Teraterm 4.73 (<https://en.osdn.jp/projects/ttssh2/releases/>)

The SPWF01S-xxxxx-xxxxxxx-RELEASE-main.ota firmware to be loaded to the Wi-Fi module must be copied into the relevant `C:\Program Files\Apache Software Foundation\Apache2.2\htdocs` web server directory for the Apache server.

The SPWF01SA Wi-Fi module allows FOTA via a single `HTTP GET` and the `FWUPDATE` command entered in the serial port terminal.

For a detailed description of this command, find the link to User manual UM1965 in [Section 6: "References"](#).

The FWUPDATE command syntax is

```
AT+S.FWUPDATE=<hostname>,<path>,<port>
```

Where:

- `<hostname>` is the target host. DNS resolvable name or IP address
- `<path&queryopts>` is the document path and optional query arguments
- `<port>` is the target host port

For example, the command for a computer with Wi-Fi IP address 192.168.0.2 is:

```
AT+S.FWUPDATE=192.168.0.2,/SPWF01S-150410-c2e37a3-RELEASE-main.ota
```

The SPWF01SA module validates the firmware image it downloads and loads it into a staging area; it then requires a reset (enter `AT+CFUN=1` in the serial port terminal) to complete the update process.

3.5.2 Configuring the Wi-Fi module

The 3D printer firmware configures the Wi-Fi module as a mini Access Point with SSID and wep key defined in `stm32_cube\Middlewares\Third_Party\Marlin\configuration.h`.

```
#define Wi-Fi_SSID "3dpserver" //Max of 32 characters
```

```
#define Wi-Fi_WEP_KEY "1122334455" //Either 10 or 26 HEX characters (0-9, A-F)
```

The Wi-Fi module can also be configured manually with AT commands described in the User manual UM1695 (see [Section 6: "References"](#)).

For example, the Wi-Fi module SSID and wep key can be changed during run time (here SSID is "my3Dap", wep key is "01234567890123456789012345" and wep key length is "0D" or 26 hexadecimals):

```
AT+S.SSIDTXT=my3Dap
```

```
AT+S.SCFG=wifi_wep_keys[0],01234567890123456789012345
```

```
AT+S.SCFG=wifi_wep_key_lens,0D
```

The mini AP IP address is 192.168.0.1 by default, but it can be changed; for example:

```
AT+S.SCFG=ip_ipaddr,192.169.0.100
```

The Wi-Fi module can also be configured as a station. For instance, the configuration for a station connection to "AndroidAP" access point using "WPA" network privacy mode and password "hdpt7892" is:

```
AT+S.SCFG=wifi_mode,1
```

```
AT+S.SCFG=wifi_priv_mode,2
```

```
AT+S.SSIDTXT=AndroidAP
```

```
AT+S.SCFG=wifi_wpa_psk_text,hdpt7892
```

3.5.3 Using the web pages

Once the user is connected by Wi-Fi to the 3D printer, by default as a station connected to 3D printer mini Access Point (SSID: 3dpserver, WEP KEY: 1122334455), the user can access the home web page by entering the 3D printer Wi-Fi IP address in a web browser (it is redirected to the `axisctrl.shtml` web page by default). All the web pages can be accessed by typing `<ip address>/<web page name>` or by following the links in the left top corner of any of the following web pages:

- `axisctrl.shtml` (see [Figure 14: "axisctrl.shtml web page"](#))
- `command.shtml` (see [Figure 15: "command.shtml web page"](#))
- `heatctrl.shtml` (see [Figure 16: "heatctrl.shtml web page"](#))
- `extructrl.shtml` (see [Figure 17: "extructrl.shtml web page"](#))
- `filemgt.shtml` (see [Figure 18: "filemgt.shtml web page"](#))
- `wifictrl.html` (see [Figure 19: "wifictrl.html web page"](#))

Figure 14: axisctrl.shtml web page

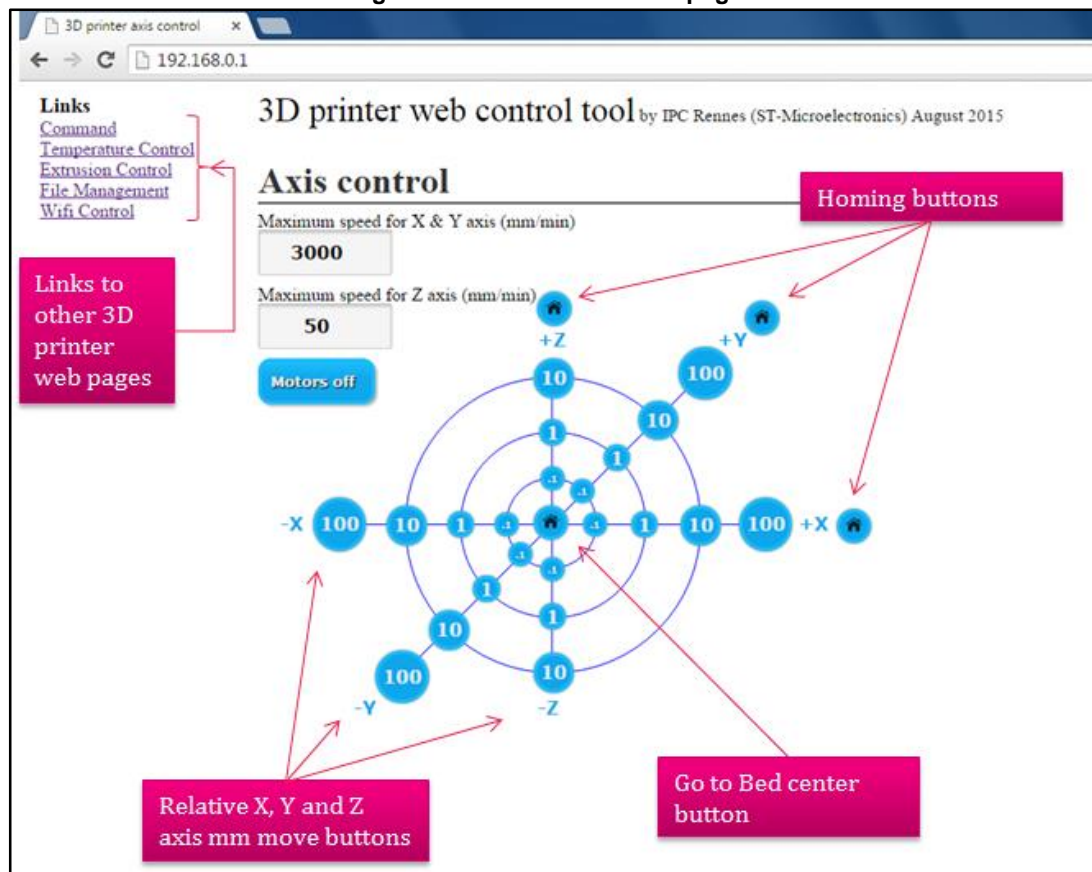


Figure 15: command.shtml web page

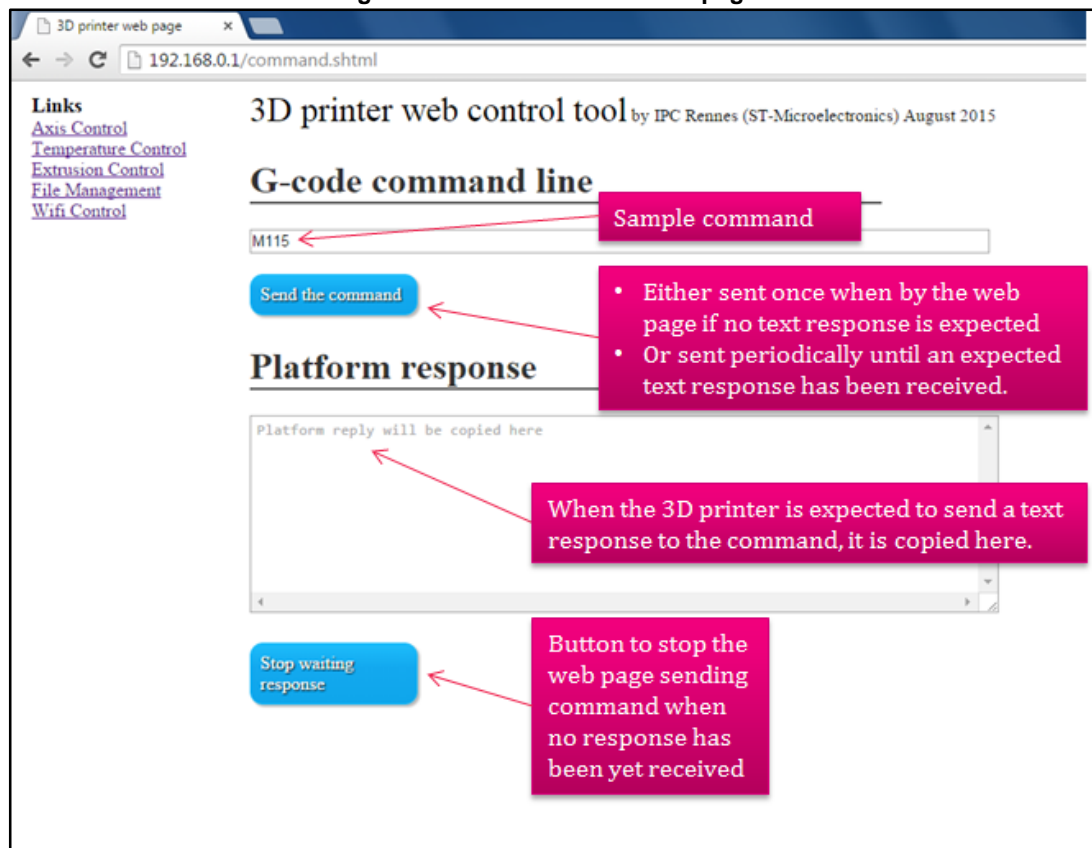


Figure 16: heatctrl.shtml web page

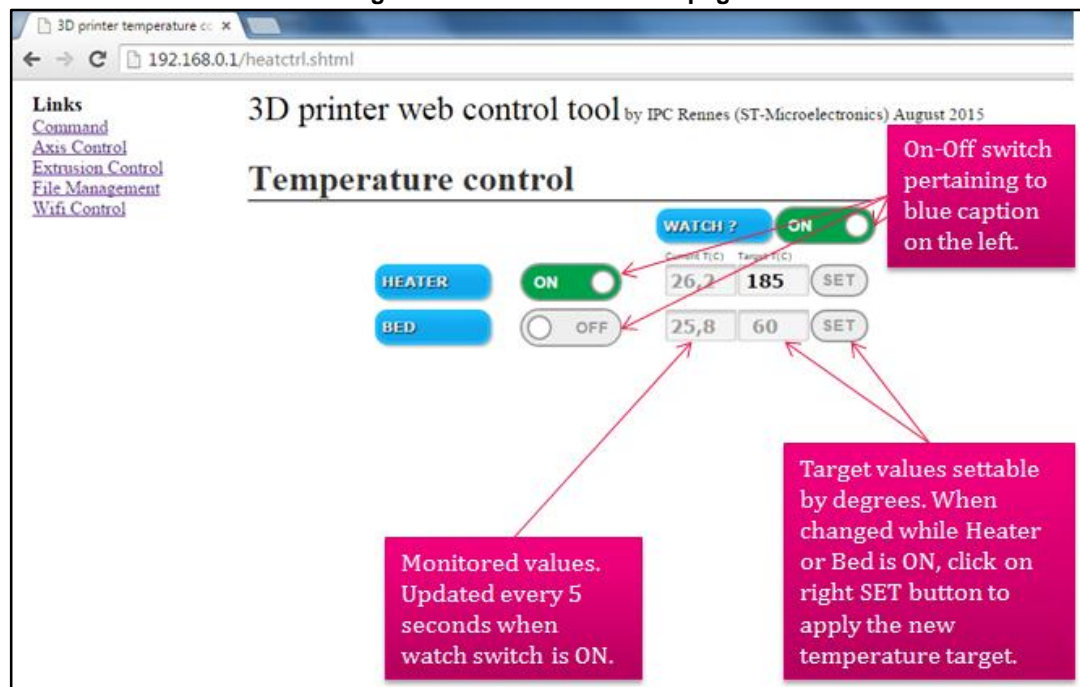


Figure 17: extructrl.shtml web page

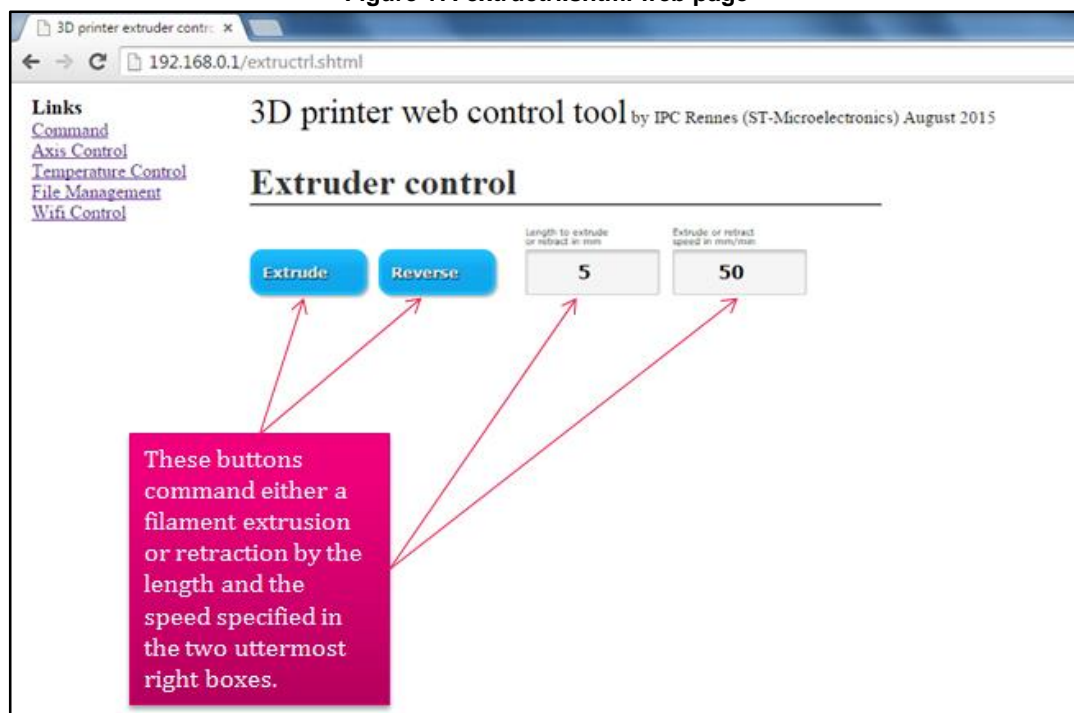


Figure 18: filemgt.shtml web page

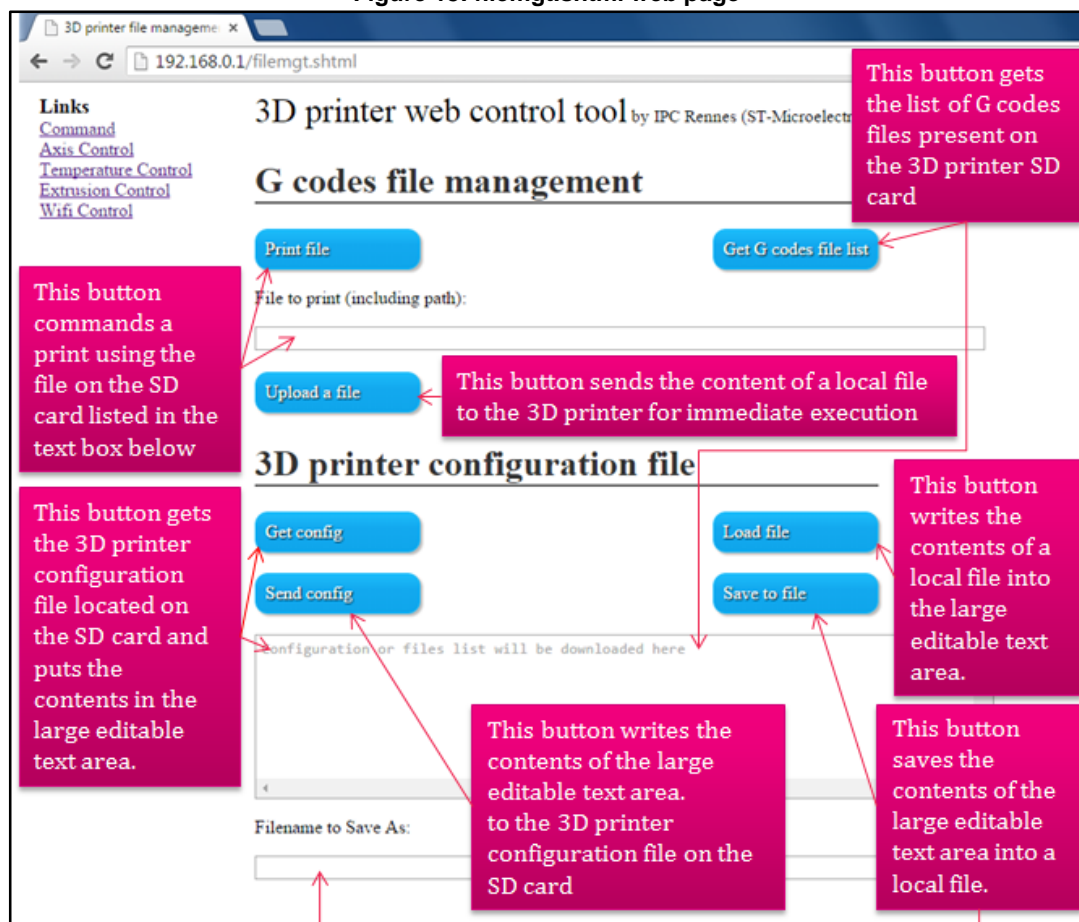
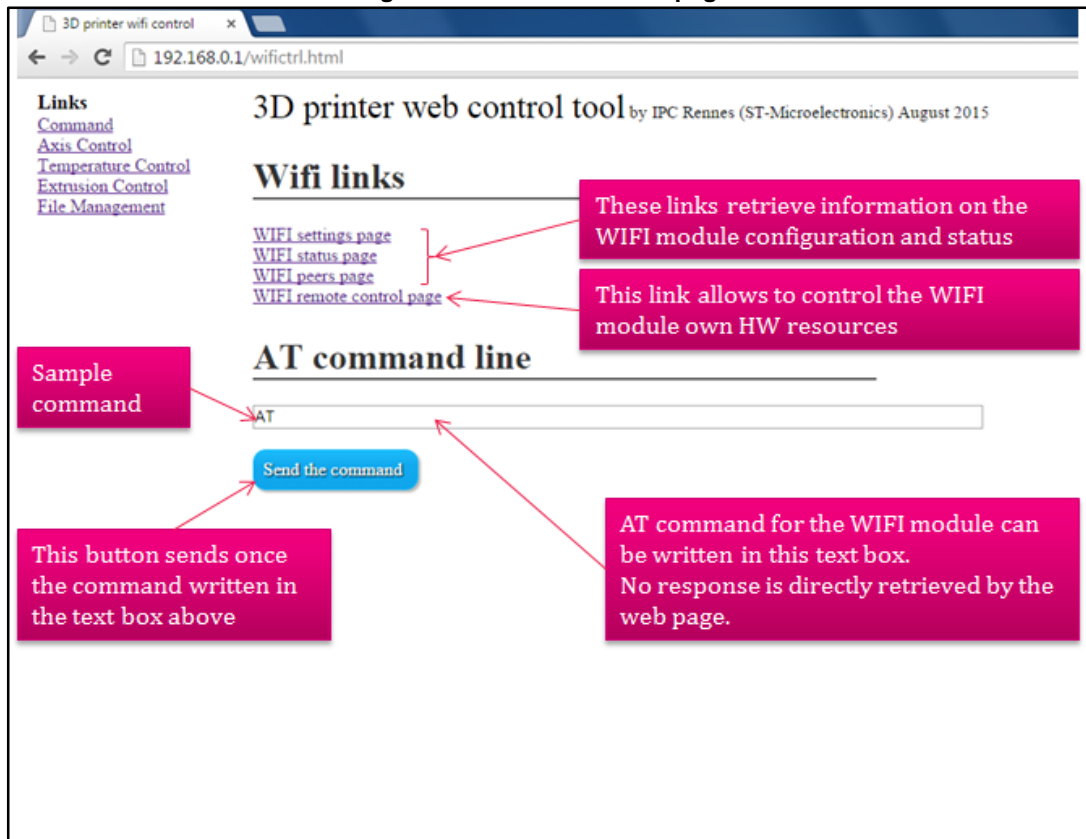


Figure 19: wifictrl.html web page



3.5.4 Customizing the web pages

The web pages presented in the sections above are given as examples and are written in html 5, javascript and css.

These pages can be modified according to the following principles:

- Design small web pages not exceeding 10 kB
- Prefer multiple small web pages to one large one
- Only flat file systems (no subdirectories) are currently supported
- If necessary, minimize the code; you can use one of the following free on-line free tools:
 - <http://kangax.github.io/html-minifier/>
 - <http://cssminifier.com/>
 - <http://jscompress.com/>
- Check that the Wi-Fi module min_heap remains above 6 k
 - click on Wi-Fi status page link to find this metric or type `AT+S.STS=min_heap` using a serial port terminal

The web page code is located in:

stm32_cube\Projects\STM32F4xx-3dPrinter\Applications\www\pages.

To update the Wi-Fi module file system with new web pages, an image file must be created using the gen.bat batch file in:

stm32_cube\Projects\STM32F4xx-3dPrinter\Applications\www.

Double-click the gen.bat file to create an outfile.img image file in the “pages” subdirectory. This image file must be copied into an accessible web server to be downloaded by the Wi-

Fi module using the command described further down. refer also to [Section 5.5: "Wi-Fi and web server"](#) for information regarding appropriate web servers.

`HTTPDFSUPDATE` is used to update the Wi-Fi module file system and change the available web pages on the http server. This command allows the creation of static files in the Wi-Fi module external flash memory for delivery by the Wi-Fi module HTTP server. The old file system is overwritten with the new image file (.img), except for the files in the Wi-Fi module STM32 internal flash memory.

The command syntax is:

```
AT+S.HTTPDFSUPDATE=<hostname>,<path>[,port]
```

Where:

`<hostname>` is the external web server; DNS resolvable name or IP address

`<path>` is the path and image file name

`<port>` is the target host port

The command for a PC with Wi-Fi IP 192.168.0.2 would be:

```
AT+S.HTTPDFSUPDATE=192.168.0.2,/fsversion01102015_001.img
```

To list the files in the file system, type `AT+S.FSL` in your serial port terminal. The first letter indicates the location in the memory (E=external flash, I=internal flash, D=internal RAM).

3.6 Serial port

3.6.1 Printing via serial port

Printing via serial port is possible with ST-LINK on the 3D printer board. To be recognized as a virtual com port in Windows, install the following ST-LINK driver:

<http://www.st.com/web/en/catalog/tools/PF260218>

Once installed, connect a USB cable between your PC and the 3D printer board ST-LINK COM port. Then, to connect to a serial port terminal, select the virtual com port associated with ST-LINK with the following settings:

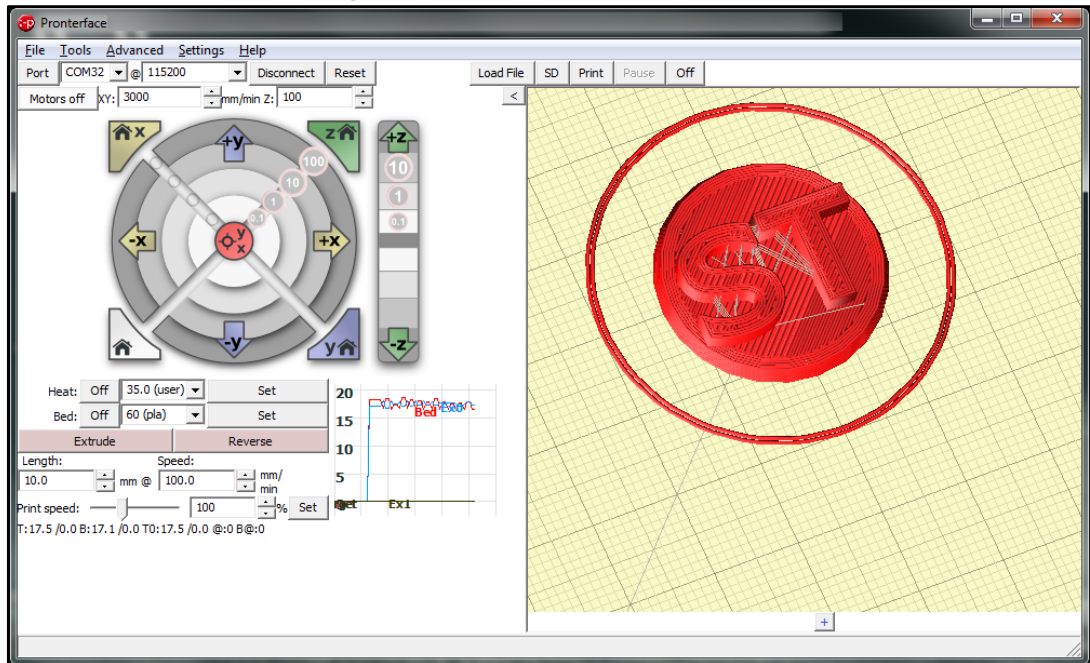
- Baud rate: 115200 (or the baud rate which is defined in file `stm32_cube\Middlewares\Third_Party\Marlin\configuration.h` if you changed the default value)
- Data: 8 bit
- Parity: None
- Stop: 1 bit
- Flow control: None
- Transmit: CR+LF
- Receive: CR
- Local echo: On

Theoretically, you can print from any serial port terminal provided you use the Marlin G-Code commands found here: <http://reprap.org/wiki/G-code>.

It is, however, easier to use a GUI to automatically send the G-Code for your commands.

You can, for example, use Pronterface: <http://www.pronterface.com/index.html>

Figure 20: Main window of Pronterface



To run a print job via UART from Pronterface:

1. Press Connect
2. Select the file to print by selecting Load File
3. Press the Print button

3.7 SD card

3.7.1 Configuration file

The default firmware requires an SD card with a configuration file which is `sys/m_cfg.g` by default, but can be changed by editing the `configuration.h` source file.

3.7.2 Printing from the SD

To print from the SD card, you can also use Pronterface to copy the targeted G-Code file on the SD card; all you have to do then is:

1. Press Connect
2. Press the SD button
3. Select the G-Code file on the SD to print

4 References

1. **UM1695** *Command set reference guide for "AT full stack" for SPWF01Sx series of Wi-Fi modules*: available on www.st.com at http://www.st.com/web/en/catalog/sense_power/FM1968/CL1976/SC1930/PF258591
2. **STSW-IDW002** *Wi-Fi Training - Hands On*: available on www.st.com at <http://www.st.com/web/en/catalog/tools/PF261605>

5 Revision history

Table 11: Document revision history

Date	Version	Changes
04-May-2016	1	Initial release.

Appendix A OpenSTM32 installation

OpenSTM32 is a plugin for Eclipse GUI and has been tested on the Kepler and Luna versions of the same. We refer to the Luna version herein.

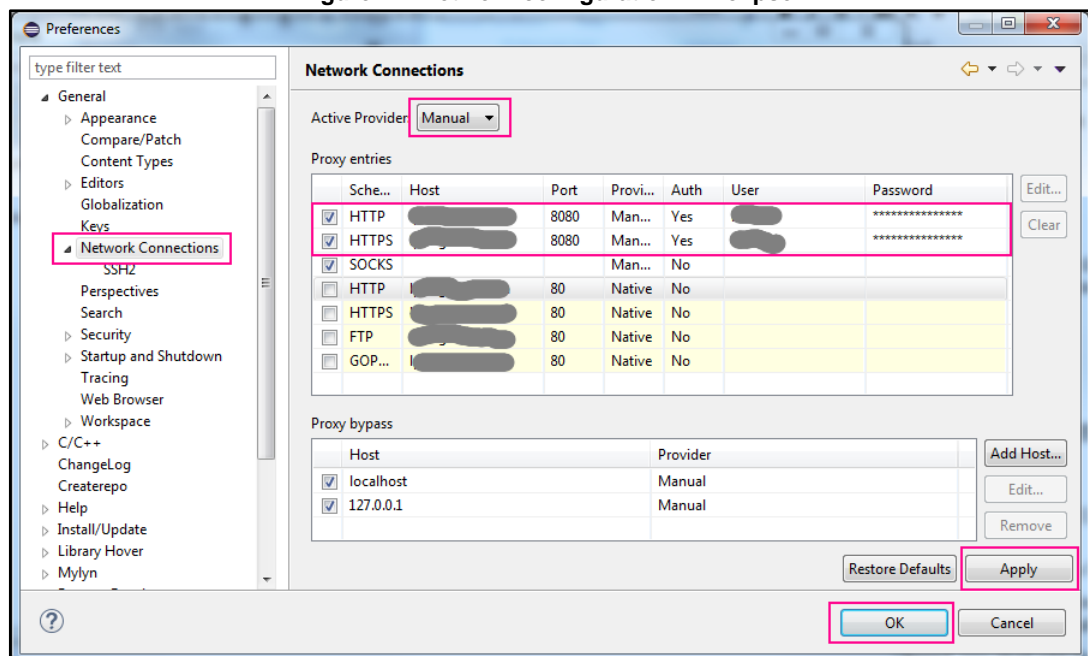
STEP 1: The OpenSTM32 multi-platform tool has been tested on Windows and on Linux OS. To install it, first install Eclipse CDT for Windows or Linux systems from <https://eclipse.org/cdt/> in the Download section.

STEP 2: Once uncompressed, add the Eclipse binary location in your PATH under Linux or create a shortcut to the Eclipse application (eclipse.exe in Windows or eclipse in Linux).

STEP 3: If you use a proxy server, you must modify the Eclipse network configuration:

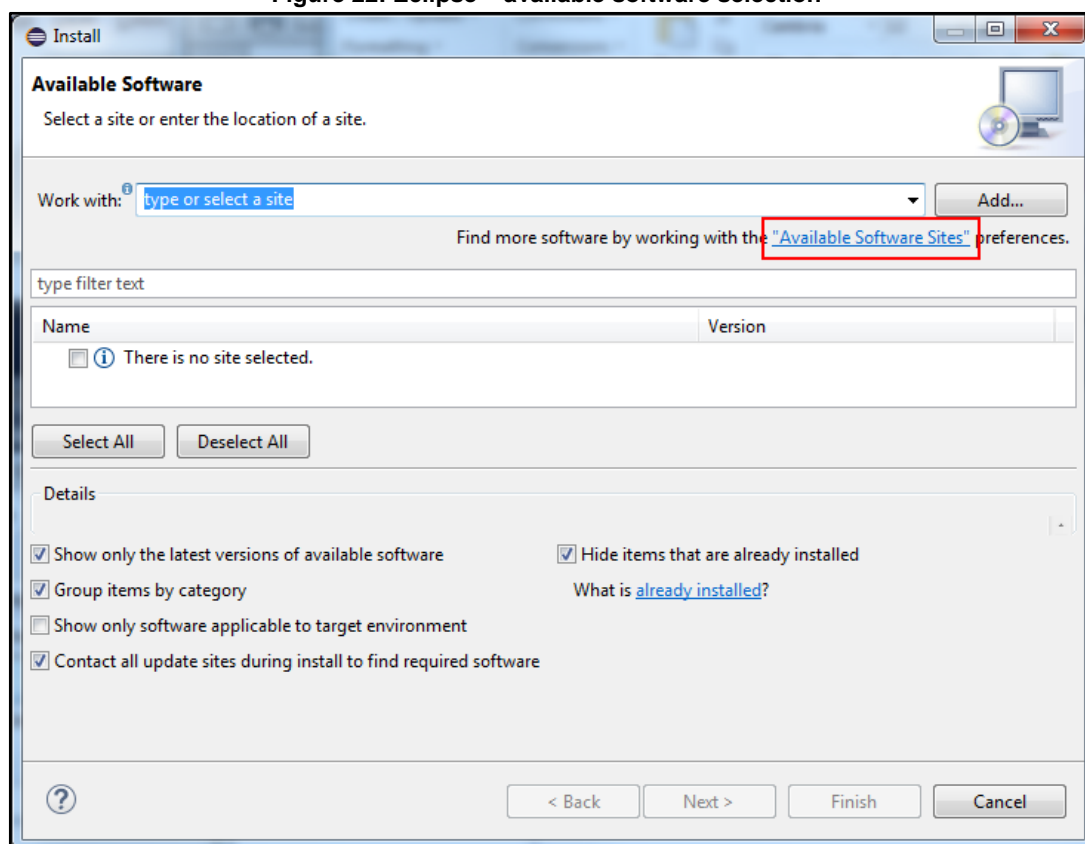
1. Open the Window > Preferences menu
2. In General > Network Connections, select manual configuration and enter your proxy address and port for http and https. Enter a user name and password if requested by your proxy.
3. Apply the changes and press ok.

Figure 21: Network configuration in Eclipse



STEP 4: To install the OpenSTM32 System Workbench plugin by ac6 in your version of Eclipse, select the Help > Install New Software menu item

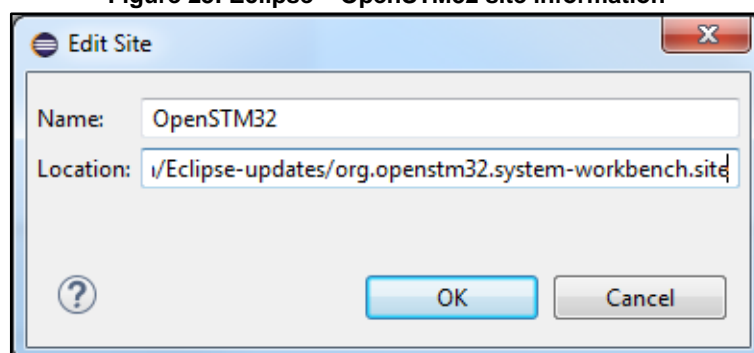
Figure 22: Eclipse – available software selection



STEP 5: Click on the “Available Software Sites” link and enter the following information:

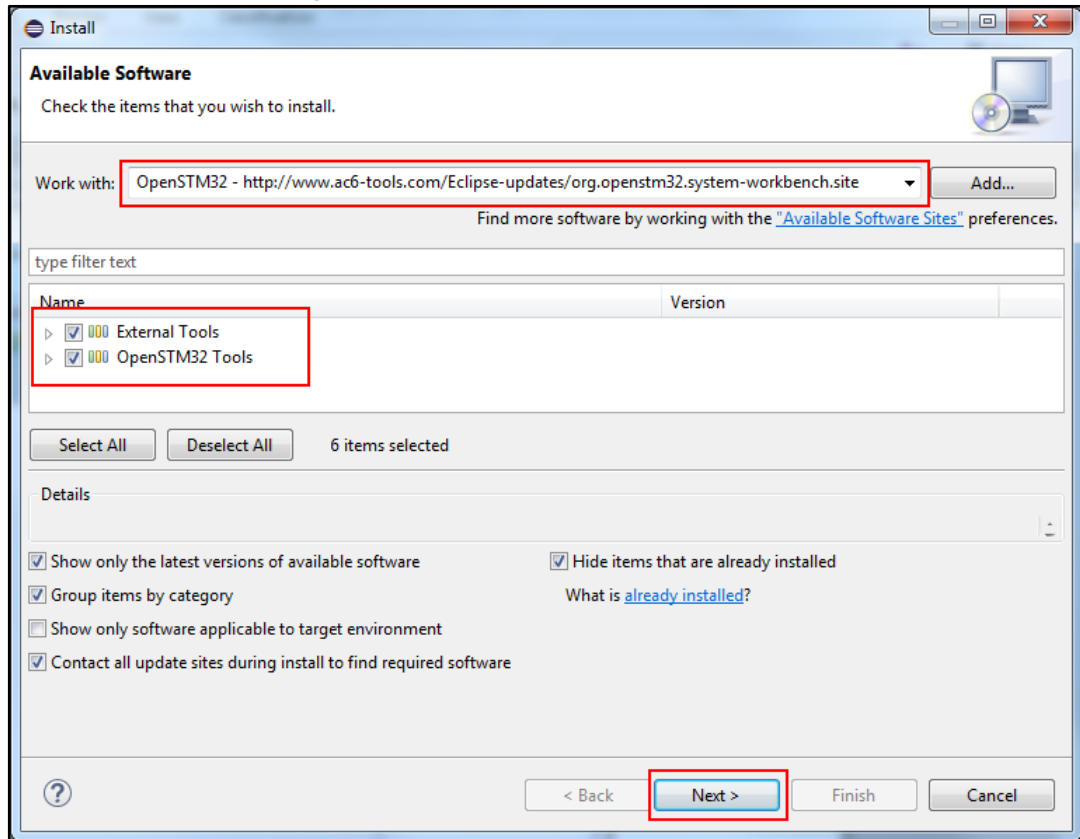
- Name: OpenSTM32
- Location: <http://www.ac6-tools.com/Eclipse-updates/org.openstm32.system-workbench.site>

Figure 23: Eclipse – OpenSTM32 site information



STEP 6: Select OpenSTM32 from the “Work with” list, check the “External Tools” and “OpenSTM32 Tools” option boxes and click on the Next button.

Figure 24: OpenSTM32 AC6 tools installation



This completes the OpenSTM32 System Workbench plugin installation.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2016 STMicroelectronics – All rights reserved